Bringing Light into the Dark - Improving Students' Black-Box Testing Competencies using Game-Design Elements

Christine Jokisch
University of Goettingen, christine.jokisch@uni-goettingen.de

Sebastian Hobert
University of Goettingen, sebastian.hobert@th-luebeck.de

Matthias Schumann
University of Goettingen, mschuma1@uni-goettingen.de

Bringing Light into the Dark - Improving Students' Black-Box Testing Competencies using Game-Design Elements

Completed Research Paper

Christine Jokisch
University of Goettingen
Platz der Goettinger Sieben 5
37073 Goettingen, Germany
christine.jokisch@uni-goettingen.de

Sebastian Hobert
University of Goettingen
Platz der Goettinger Sieben 5
37073 Goettingen, Germany
sebastian.hobert@uni-goettingen.de

Matthias Schumann
Platz der Goettinger Sieben 5
37073 Goettingen, Germany
mschuma1@uni-goettingen.de

Abstract

As software becomes increasingly complex, there is a growing need to enhance quality assurance in software engineering. However, the lack of qualified human resources is a barrier to performing software testing activities in software companies. At the same time, software testing can be considered a tedious task and is often not done at the necessary level of detail, e.g., designing test cases. However, it is crucial for novice programmers and testers to acquire and improve their testing competencies, and to utilize testing techniques, e.g., black-box testing. Teaching software testing is often based on theoretical instructions, resulting in limited practical experience. As a result, students may not develop the necessary testing mindset, highlighting the need for more extensive software testing education. To address this issue, this paper utilizes a design science research approach to implement a gamified learning system that promotes black-box testing competencies with empirical insights from a field test.

Keywords: Software Testing Education, Design Science Research, Testing Competencies

Introduction

As information systems become complex, there is a growing need to prioritize quality assurance in software engineering (Costa & Oliveira, 2019). An essential part of ensuring product quality is adequate testing. By running programs or models with specific inputs, software testing verifies that the software behaves as expected, thus, enhancing the software quality (Valle et al., 2020). Identifying and addressing existing defects before the customer receives the product can increase its reliability (Valle et al., 2020).

Although software testing is recognized as a crucial activity in assuring the quality of software products, the industry has reported a shortage of specialized and qualified professionals in this field. Indeed, both senior developers and novice programmers have difficulties applying test techniques, criteria, and testing tools to entire programs or functionalities (Scatalon et al., 2017; Valle et al., 2020). The lack of competencies may be linked to a deficiency in software testing education and a lack of motivation in the workplace (Valle et al., 2020). Even though, software testing and quality assurance are considered essential parts of software engineering processes, it does not intake an equal part of software testing education (Hynninen et al., 2019), i.e., software testing often occupies a less significant role in industry and academia (Krutz et al., 2014). This highlights the need for academia to prepare novice programmers and engineers for future careers in this field (Costa & Oliveira, 2019; Sánchez-Gordón & Moreno, 2014).

To enhance student's ability to perform testing activities, institutions are taking steps to promote and provide training opportunities (Paschoal et al., 2019). However, some institutions or testing courses only provide theoretical instructions on testing concepts, with little or no practical experience, resulting in students not developing the habit or awareness of testing (Hynninen et al., 2019). Thus, there is a need to use educational strategies to teach software testing practically, beyond the traditional and theoretical instructions (Paschoal et al., 2019).

There is a wide range of techniques within software testing. One testing technique that received less attention is called black-box testing. As the black-box testing does not reveal the internal source code of the software under test, students need to design test cases based on the software component specifications (Sharif & Hemmati, 2018). Those test cases can be designed both manually and automatically (Sharif & Hemmati, 2018). Although automated testing has become popular with the development of scripting languages and testing tools, manual testing remains necessary for a significant number of tests, e.g., end-to-end data checks of data transfer and values (Sharif & Hemmati, 2018). For this, critical thinking and practical knowledge are required, e.g., to discover flaky tests, oracle problems, or edge cases. However, manual testing is often perceived as tedious and monotonous by developers or test engineers and is rarely done at the required granularity, resulting in poor project quality (Garousi et al., 2020). As a result, software testing is considered an unmotivating task, which is time-, cost-, and effort-consuming (Jesus et al., 2018).

For this reason, motivation and discipline have become crucial elements of software testing education (Garousi et al., 2020). The use of game-design elements aims to increase motivation and engagement for improving education and training processes (Valle et al., 2020). In this study, we will address the problem of inadequate training in functional testing and provide a practical solution that supports students' learning process by using game-design elements. Finally, this paper will present a software testing learning system to enhance the functional testing competencies of novice programmers. To achieve our goal, we follow a design science research approach based on Hevner (2007) to answer the following research question:

> **RQ:** *How should a motivational information system be designed in order to teach students black-box testing competencies?*

First, we will provide a brief overview of the fundamentals of software testing and gamification. Then we will outline the research methodology. Following this, we will present two design iterations, which were developed through six consecutive steps. In the final stage, we will present the results of a field test, consisting of pre- and post-tests conducted with undergraduate students in an information systems course. Upon the completion of the evaluation and a field test, we will present design principles. Finally, we will conclude by highlighting the limitations of the study and future research directions.

Theoretical Background and Related Research

Software Testing in Software Engineering

Software testing is an integral part of software engineering processes since it examines whether the software works accurately according to the system specifications and fulfills user requirements (Hynninen et al., 2019; Thakur & Sharma, 2018). In particular, software testing is the process of running a program or system and aims to find errors by enabling to analyze of whether such software behaves as expected (Thakur & Sharma, 2018). However, a detailed analysis of the system can detect errors and defects which can be fixed by debugging (Valle et al., 2020).

Moreover, within the software development life cycle, *validation* and *verification* are two processes used to test the software (Kaprocki et al., 2015; Thakur & Sharma, 2018). Firstly, *software validation* refers to checking software functionality to determine if it meets the customer's requirements. It ensures the completion of the software based on the user requirements and checks if the right product is built (Thakur & Sharma, 2018). However, *software verification* is a process of ensuring that the software product meets all the necessary business requirements. The verification checks if the product is built right. The verification process focuses on the design and system specifications (Thakur & Sharma, 2018).

Figure 1 illustrates the testing activities within the testing process based on Alhroob et al. (2012). These activities also encompass the fundamental core competencies of the black-box software testing process: test design, test execution and test analysis. Firstly, the testing process starts with the design process by designing test cases that contain conditions for inputs and likely output of a system (mark 1). The test cases are then prepared based on the testing strategy, e.g., techniques such as black-box or white-box techniques (Alhroob et al., 2012). Afterward, each test case and its test data are executed on the software under test (SUT) in the execution phase. After the test case execution, the process will deliver an actual result as output (mark 2) and be used to compare and analyze the expected result from the test case design (mark 3). A software or test case bug may be detected when the actual result does not coincide with the expected result. Thus, the test engineer needs to report the result (Jain & Kaluri, 2015).

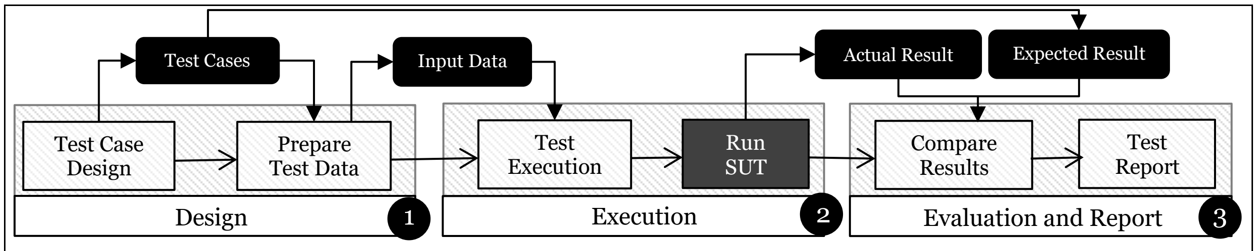


Figure 1. Software Testing Process based on Alhroob et al. (2012)

In summary, there are three different tests: *black-box testing*, *white-box testing,* and *grey-box testing* (Hooda & Chhillar, 2015). While grey- and white-box testing depend and varies on the internal source code of a software, black-box testing is independent of technical understanding and programming language. Therefore, it is more appropriate for novice developers. For this reason, we focus on black-box testing.

Software testing principles have been identified as one of the areas in software engineering, that should be integrated early in the curriculum. In this regard, approaches for teaching black-box competencies have been developed in prior research: Elbaum et al. (2007) introduced as one of the first, a web-based gamified learning environment *Bug Hunt*, interactive and engaging methods to teach software testing principles, specifically both white-box and black-box testing lessons. Yujian and Clarke (2016) proposed WReSTT-CyLE (Web-Based Repository of Software Testing Tutorials - a Cyberlearning Environment), which includes social features and testing objects to enhance students' conceptual understanding and practical skills in software testing by providing tutorials and testing objects. However, no detailed description of the testing (Paschoal et al., 2019) objects in the WReSTT-CyLE was given. Yet, these learning environments are limited to 1) creating simple test cases with few numerical values instead of considering multi-step test cases with non-numerical inputs, and 2) focusing on one aspect within the software testing process (design). Thus, competencies, e.g., during the execution of test cases are not promoted. This study aims to develop a learning environment that promotes the holistic understanding and application of black-box testing competencies using different testing techniques. Thus, we aim to overcome limitations of existing approaches in terms of the test case complexity and scope.

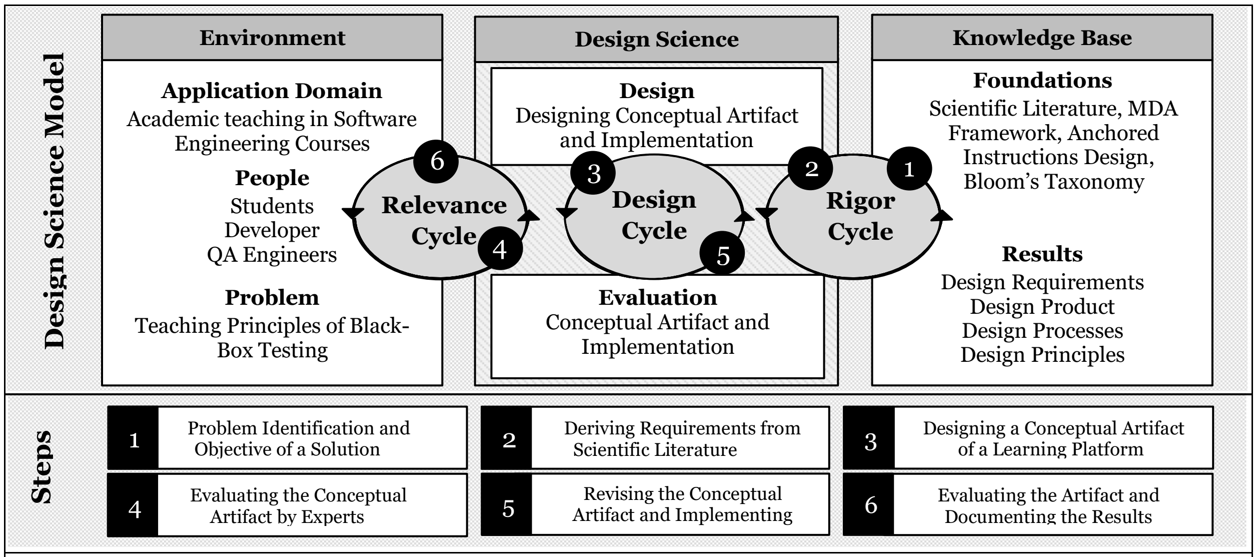
Gamification

In recent years gamification has been increasingly used for innovative teaching approaches by utilizing a gameful design for inducing experiences from games to support activities to increase students' motivation and engagement (Deterding et al., 2011). However, according to Kapp (2012), "*Gamification is using game-based mechanics, aesthetics, and game-thinking to engage people, motivate action, promote learning, and solve problems*." Thus, the positive outcomes and behavior are the results of the process of gamification. However, by creating a motivational learning environment, we aim to improve the user experience and make the learning process more enjoyable. Moreover, depending on the choice of game mechanics, qualitative and quantitative performance results can be influenced (Sailer, 2016). Thus, we believe that gamification can improve the quality of designed test cases.

Based on this, game-design elements can be derived using the mechanics, dynamics, and aesthetics (MDA) framework (Hunicke et al., 2004). The MDA Framework enables a user-centered game-design model to ground the design process of a gamified learning system (Hunicke et al., 2004). Thus, it provides a specific user-centered focus on the user's needs and learning objectives. The *mechanics* describe the components of a game at the level of data representations and algorithms, e.g., points, badges, or leaderboards. However, the *dynamics* focus on the run-time behavior and the input and interactions by users, e.g., progression, collection, or cooperation. Lastly, aesthetics are emotions that reflect the feelings evoked by the mechanics and dynamics. In addition, a variety of elements could help address different psychological needs, e.g., autonomy, competence, and social relatedness based on the self-determination theory (Deci & Ryan, 2012). Thus, in this paper, we will use the MDA framework to address the three psychological needs of the self-determination theory (SDT).

Research Design

To address our research goal of designing a gamified learning platform to improve students' testing competencies of black-box testing, we use the design science research approach based on Hevner (2007).



Design Science Model
Environment
Application Domain
Academic teaching in Software Engineering Courses
People
Students
Developer
QA Engineers
Problem
Teaching Principles of Black-Box Testing
Design Science
Design
Designing Conceptual Artifact and Implementation
Evaluation
Conceptual Artifact and Implementation
Knowledge Base
Foundations
Scientific Literature, MDA Framework, Anchored Instructions Design, Bloom's Taxonomy
Results
Design Requirements
Design Product
Design Processes
Design Principles
Relevance Cycle
Design Cycle
Rigor Cycle
1
2
3
4
5
6
Steps
1 Problem Identification and Objective of a Solution
2 Deriving Requirements from Scientific Literature
3 Designing a Conceptual Artifact of a Learning Platform
4 Evaluating the Conceptual Artifact by Experts
5 Revising the Conceptual Artifact and Implementing
6 Evaluating the Artifact and Documenting the Results

Figure 2. Design Science Research Model

As illustrated in Figure 2, we used the design science research approach by Hevner (2007) and applied six sequential research steps based on the relevance, rigor, and the design cycles. The first step of the design science research process started by specifying the problem and presenting the objective of a solution based on current research (**step 1**). Afterward, we conducted a literature review following the methodology by Webster and Watson (2002). Based on the literature review, we identified functional and game-based requirements for testing factual concepts (**step 2**). Subsequently, the requirements form the fundamentals for the conceptual artifact (**step 3**). Finally, we evaluated the concept through semi-structured interviews conducted with 21 experts, including developers, test engineers, or quality assurance engineers, who possess a minimum of two years' experience in software testing (**step 4**). The subsequent design phase consists of revising the concept based on the derived feedback from the experts and implementing the concept into a software system (**step 5**). The design cycle ends with an evaluation of the implemented gamified learning platform of black-box testing followed by the documentation of the design knowledge based on Gregor et al. (2020) formulation of design principles (**step 6**).

Designing and Evaluating Gamified Black-box Testing System

In the following section, we will summarize the results of the two design iterations based on six consecutive steps and present the design of the learning application called *iTest app*. During the design process, we adopt the Anchored Instruction Design approach (Cognition and Technology Group, 1993) and Bloom's revised taxonomy (Anderson & Krathwohl, 2001; Bloom, 1956). It provides an educational solution, e.g., information systems (IS) courses, to teach the principles of black-box testing competencies and make students aware of their actions and related black-box testing concepts.

Problem Identification and Objective of a Solution

The foundation of our problem identification is based on the low attention derived from both the educational context and current research regarding learning-based approaches to improve black-box competencies (Ottfutt et al., 2011). First, the manual design and execution of test cases play a subordinate role in current research. Instead, due to an increasing number of test cases and execution frequency, the automation of test cases is coming to the fore (Haas et al. 2021). Thus, significant research efforts have been made toward optimizing automated testing, e.g., regression test optimization. However, only a few research efforts attempt to transfer test techniques, e.g., the boundary value analysis to manual black-box testing (Porto et al., 2021). Nevertheless, there is still a considerable need for manual design and execution of test cases as both techniques complement each other (Haas et al., 2021). Thus, there is a need for prospective software testers or testers to develop a wide range of knowledge in using different testing techniques.

However, in most cases, students in information technology, software engineering, computer science, or information systems are either not introduced to testing concepts or are often lectured about testing theoretically (Santos et al., 2021). Focusing only on factual knowledge limits the learning outcome, which are the basic cognitive levels of educational learning objectives according to Bloom's taxonomy (Bloom, 1956). According to the taxonomy, the cognitive levels are remembering, understanding, applying, analyzing, evaluating, and creating. Though remembering, understanding, and applying are the basic cognitive levels, analyzing, evaluating, and creating refer to the higher level of the cognitive level. However, learning factual knowledge is insufficient to fulfill employees' tasks, e.g., as a test engineer in a company. Indeed, students get little practical training in designing test cases based on functionalities or requirements specifications which are the foundations of black-box testing. Consequently, it may lead to detrimental habits that are difficult to change (Sharif & Hemmati, 2018).

One of the drawbacks of traditional learning environments is that they often fail to engage students sufficiently as software testing tends to be less captivating. Furthermore, the latest research conducted by Blanco et al. (2023) concluded that students who participated in the gamified course outperformed the control group. In addition, they noted that the key to success is the gamified experience design. This can be transposed onto the scenario of black-box testing. As gamification promotes interactivity, it aligns with the need for hands-on exploration in the realm of software systems, especially in black-box testing scenarios.

Thus, this paper aims to provide a comprehensive educational experience on the design, execution, evaluation, and reporting of test cases. By using an accessible web application as a supplementary learning tool, lecturers can facilitate knowledge transfer and students by enabling them to acquire practical insights into software testing methods compared to non-system based approaches. The primary novelty of our approach lies in the investigation of the impact of gamification on the teaching of functional software testing, as well as the introduction of practical testing objects to facilitate the learning of black-box testing.

In general, the main objectives of the design science research process is to:

(1) Enhance the quality of written test cases by integrating game-design elements.
(2) Encourage students to manually execute and review test cases.
(3) Foster reflection on test cases created by other students.
(4) Enhance knowledge of black-box testing by integrating practical-oriented testing objects.

By incorporating the concept of black-box testing into educational programs, the goal is to elevate the understanding and importance of software testing, as well as stimulate interest and motivation towards this area. Therefore, early exposure and education in software testing can potentially contribute to the success of software projects in the workplace.

Deriving Requirements from Scientific Literature

Based on scientific literature and the software testing process, we identified the key requirements necessary for designing a conceptual artifact during the first iteration of our study. We applied a selective literature analysis based on Webster and Watson (2002) using the search term: *("Gami*") AND ("Black*box Testing"*

OR "Functional Testing") AND ("Manual"). We identified requirements for designing a conceptual artifact on seven databases (e.g., ACM Digital Library or IEEE) and differentiated the requirements between (1) core functional requirements of the learning system, and (2) gamification-based requirements.

Core Functional Requirements (R_F)

Finally, we identified five functional requirements forming the base of the learning system (See Table 1).

#	Requirements	Bloom's Taxonomy	Exemplary References
R_{F1}	Providing video-based learning material to gain theoretical knowledge about functional testing.	Remember	Cognition and Technology Group (1993); Paschoal et al. (2019); Elbaum et al. (2007)
R_{F2}	Testing theoretical knowledge on functional testing by a quiz.	Understand	Valle et al. (2020); Jesus et al. (2019); Yujian and Clarke (2016)
R_{F3}	Designing test cases considering different input combinations based on different feasible testing objects and requirements.	Apply	Valle et al. (2020); Clarke et al. (2014); Elbaum et al. (2007)
R_{F4}	Run tests manually and analyze results with the expected results.	Analyze	Scatalon et al. (2017)
R_{F5}	Evaluating and reporting created test cases with options to take appropriate actions if necessary.	Evaluate	Scatalon et al. (2017)

Table 1. Core Functional Requirements

To differentiate between exercises, we utilized Bloom's Taxonomy (Bloom, 1956), which offers various options for determining the tasks on the learning platform. The platform provides students with the opportunity to (1) remember fundamental facts about functional testing, (2) understand the use of functional testing, (3) apply knowledge on various realistic objects, (4) analyze, and (5) evaluate test cases.

To avoid passive and decontextualized knowledge transfer, we adopted the anchored instruction design approach based on video material (Paschoal et al., 2019). This approach requires designing a technologically-supported learning environment that demands and encourages active, problem-oriented learning (Cognition and Technology Group, 1993). Thus, the learning application provides *video-based learning materials* ($\rightarrow R_{F1}$) as anchors and offers independent and explorative problem recognition, definition, and solving. Moreover, the learning platform includes also an aligned *quiz* ($\rightarrow R_{F2}$) consisting of questions about functional testing to strengthen the knowledge about testing (Jesus et al., 2019; Valle et al., 2020; Yujian & Clarke, 2016). Additionally, a pivotal element of the proposed learning platform is the utilization of *testing objects* ($\rightarrow R_{F3}$), which serve as compact representations of various functionalities. The integration of testing objects, e.g., real functional objects, facilitates the transition from theoretical concepts to practical applications, thus, enabling a more profound level of learning (Yujian & Clarke, 2016). As a result, the *execution of test cases* ($\rightarrow R_{F4}$) becomes a straightforward task, in which students are expected to apply the test cases to the associated testing object and analyze the actual results with the expected results. As a last identified requirement, the test cases need to be *evaluated* ($\rightarrow R_{F5}$) based on the manually executed tests. This involves critical reflection on the designed test cases, evaluating their quality, providing feedback, and taking appropriate actions, e.g., submitting a report if the test case is insufficiently designed.

Gamification-based Requirements (R_G)

By incorporating game-design elements into the gamified learning system, we aim to positively impact the learning process as it can address different psychological needs based on the self-determination theory (Deci & Ryan, 2012). For this, we identified eight game-design elements.

Profile. Using a social *profile* ($\rightarrow R_{G1}$) that summarizes and records all associated mechanics data can draw attention to individual testing activities and values (Costa & Oliveira, 2019; Kris & Heider, 2020). As software testing consists of different tasks (design, execution, evaluation) the data can be published on the profile, leading to the dynamic *expression* and the need for *competence* and *social relatedness*.

Points. By collecting *points* ($\rightarrow R_{G2}$), e.g., experience points or after completing exercises, one's current progress can be represented (Clarke et al., 2014). Moreover, according to Valle et al. (2020), points maintain users' attention for conducting tasks proposed since it promotes *competition* among the students. As a result, points can increase the user's engagement and generate peer pressure to keep playing (Kris & Heider, 2020), thus supporting the need for *competence* according to the Self-Determination Theory.

Level. Based on collected points, a user's *level* (→R_{G3}) represents the visual *progression* of one's current knowledge state (Jesus et al., 2019). Moreover, the level system can cause a peer-pressure which represents the level of testing skills (Parizi, 2016). However, it can also enhance the user's sense of *achievement* upon completing tasks and reaching a higher level (Valle et al. 2020), thus promoting the need for *competence*.

Progress bars. Using *progress bars* (→R_{G4}) for exercises, students are given control over their learning process and monitor their testing activities, leading to the dynamic *progression* (Yujian & Clarke, 2016). Moreover, it provides individual *feedback* and encourages students to set their own goals and develop effective learning strategies (Yujian & Clarke, 2016), thereby fostering the need for autonomy.

Quests. *Quests* (→R_{G5}) are tasks that serve as a real-time feedback mechanism, providing students with updates on their *progress* toward achieving learning objectives (Jesus et al., 2019). Moreover, quests are designed to represent a journey of obstacles that students must overcome (Garcia et al., 2017; Jesus et al., 2018), e.g., the difficulty of designing test cases based on testing objects can increase, which could lead to the feeling of *competence* when students overcome these obstacles.

Badges. *Badges* (→R_{G6}) are a visual representation of achievements or milestones that can trigger the dynamic *collection*. They are obtained for completing specific activities, e.g., discovering edge cases. The representation can foster the need for *competence* by reflecting the user's performance (Garcia et al., 2017). Thus, badges act as a feedback mechanism, showing the user's *progress* and accomplishments.

Leaderboard. The *leaderboard* (→R_{G7}) is a feedback mechanism that enables users to monitor their ranking compared to others in the learning environment (Garcia et al., 2017), e.g., based on cumulated experience points (Yujian & Clarke, 2016). While leaderboards provide motivational *feedback*, they can also promote dynamic *competition* as users strive to outperform each other (Jesus et al., 2019). To address the need for *autonomy*, it is advisable to provide an opt-in option for leaderboards (Jesus et al., 2019).

Social Rating. For ensuring the quality of activities, *social ratings* (→R_{G8}), e.g., star rating can be used on designed test cases, thus, give them more freedom to assess test cases (Porto et al., 2021). Based on Kapil Singi (2020), a star taking (0-5 stars scale) can be used to foster the dynamic *competition* among students since the rating is an indicator of one's performance and thus, address the psychological needs for *autonomy* and *competence*.

First Iteration: Designing the Conceptual Artifact

According to the third step of the design science research approach, we designed a conceptual artifact based on the derived requirements from the knowledge base. Figure 3 illustrates the overall concept.

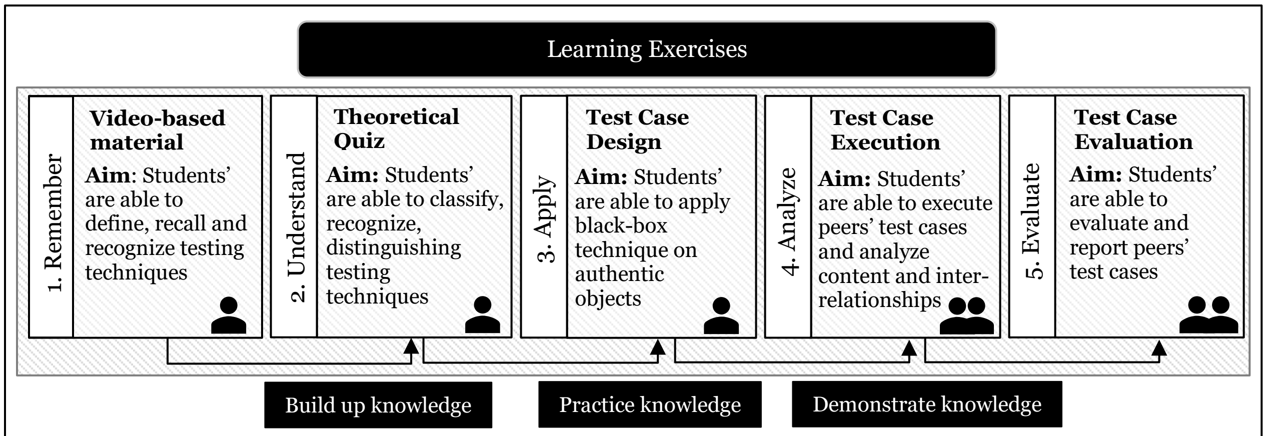
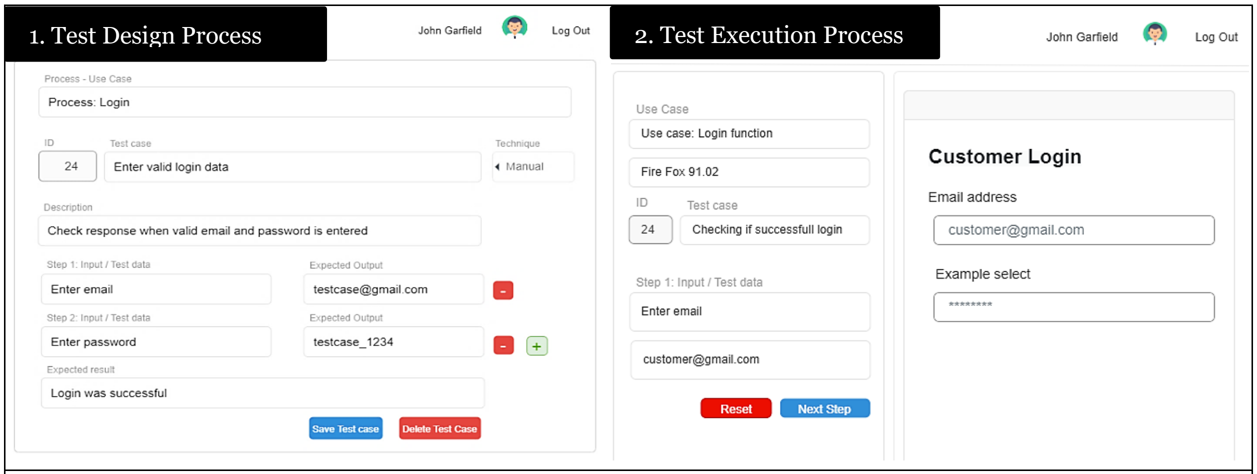

Learning Exercises
1. Remember
Video-based material
Aim: Students' are able to define, recall and recognize testing techniques
2. Understand
Theoretical Quiz
Aim: Students' are able to classify, recognize, distinguishing testing techniques
3. Apply
Test Case Design
Aim: Students' are able to apply black-box technique on authentic objects
4. Analyze
Test Case Execution
Aim: Students' are able to execute peers' test cases and analyze content and inter-relationships
5. Evaluate
Test Case Evaluation
Aim: Students' are able to evaluate and report peers' test cases
Build up knowledge
Practice knowledge
Demonstrate knowledge

Figure 3. Overall Concept of the Learning System

The proposed concept comprises five integrated components that are designed as exercises within the learning application, which are aligned with the core functional requirements. To comprehend the concept, we developed visual mockups that illustrate the functional requirements and incorporate game-design elements, as outlined before. The mockups depict an overview of learning sessions that guide the user through the main components of functional requirements in a sequential manner. These components include: acquiring knowledge through the video-based learning material, completing a quiz, and designing, executing, and evaluating test cases.



Process - Use Case: Process: Login
ID: 24
Test case: Enter valid login data
Technique: Manual
Description: Check response when valid email and password is entered
Step 1: Input / Test data: Enter email
Expected Output: testcase@gmail.com
Step 2: Input / Test data: Enter password
Expected Output: testcase_1234
Expected result: Login was successful

Use Case: Use case: Login function
Fire Fox 91.02
ID: 24
Test case: Checking if successfull login
Step 1: Input / Test data: Enter email
customer@gmail.com

Customer Login
Email address: customer@gmail.com
Example select: ********

Figure 4. The Conception of Test Design and Execution Process

To enhance students' learning experience, the motivational information system is designed with several exercises. The first exercise refers to *video-based* learning (→R_{F1}) aimed at building knowledge through observation. The student's progress in each exercise is supported by a horizontal *progress bar* (→R_{G4}) and associated *quests* (→R_{G5}) as feedback mechanisms to track and show the current achieved goals. Moreover, the student will receive *points* (→R_{G2}) and *badges* (→R_{G6}) if an exercise is completed. The points are part of a level system (→R_{G3}) that indicates the student's level of knowledge. Additionally, students can share their opinions and rate the exercise using a rating system.

In the subsequent exercise, the students' theoretical knowledge is evaluated through a *quiz* (→R_{F2}) about functional testing. Upon answering all questions correctly, the exercise leads to practical exercises. The practical exercises consist of *designing a test case* (→R_{F3}) based on a predefined form and a testing object, e.g., a login system. The predefined form consists of input fields about the overall test suite, a short description of the test case, the technique used, and dynamically expandable test steps. The test steps are composed of a category, description, and input data. Subsequently, an expected result is also to be specified. After designing test cases, students can *execute a test case* (→R_{F4}) created by another student. During the manual execution, the underlying testing object or functionality is displayed, allowing all steps to be run through consecutively. If any conspicuousness is found, the test cases can be reported which is accompanied by an *evaluation of a test case* (→R_{F5}) using a *star rating system* (→R_{G8}). Additionally, qualitative feedback can be given. Finally, the user profile summarizes the achievements, progress, and results, e.g., a *leaderboard* (→R_{G7}) and a *level system* (→R_{G3}) based on received points.

First Iteration: Evaluating the Conceptual Artifact by Experts

After designing a conceptual artifact based on the requirements from scientific literature, we conducted 22 guided interviews with experts, including developers, test engineers, and test managers with knowledge about functional testing to gather feedback on the conceptual artifact. The interviews were structured in three blocks: The first block consisted of an introduction to the project's motivation. Then, in the second block, challenges and solutions were questioned. In the third block, the mockups were presented successively. For each mockup, the following questions were asked:

(1) *missing functionalities*,
(2) *improvements to existing functionalities*,
(3) *missing clarity or traceability on functionalities*,
(4) *missing motivational elements*, and lastly,
(5) *improvements of motivational elements*.

An excerpt of the result is shown in Table 2. We considered results where the experts' feedback showed agreement.

Function	Description	Interview Reference
Introduction	Introduce functions by using an onboarding system.	Expert 1, 4, 12
Test Case Design	Adding Preconditions to the test case design.	Expert 8, 3, 5
	Reduce Information fields on test case design.	Expert 7, 9, 14
Test Case Execution	Possibility to report each step.	Expert 2, 14
Test Case Evaluation	Comment function when completing the test execution.	Expert 3, 4, 5, 9, 11, 13
	Add attachments when completing the execution.	Expert 10, 2, 20
Profile	Adding an option about publishing profile information.	Expert 5, 18
Leaderboard	Reset the leaderboard based on recently created test cases.	Expert 8, 19
	Adding unranked status when reaching a lower rank.	Expert 3, 4, 5, 8
Points	Allocation of points based on test case complexity, e.g., steps, received evaluations, and when executing test cases manually.	Expert 2, 3, 6, 7, 10, 14

Table 2. Summarized Feedback on the Conceptual Artifact

General Feedback. As an additional function, three experts suggested the implementation of an *onboarding system* for first-time usage to provide users with a step-by-step guide, "*We should guide people and say, 'Once you've done this, write it down here.*'" (**Exp-4**) Moreover, regarding the *test case design* process, three experts claim a notable need to add *preconditions*, which are understood as a series of steps that must be completed before adding new steps to a test case. Further, three experts suggested reducing the number of *information fields*, e.g., ID or browser type, to prevent an information overload.

Process-related Feedback. Regarding the *test case execution* process, two experts suggested implementing a *feedback* function when executing each step of a test case, "*This 'passed' or 'failed' must be possible for each step, so one can say, 'Okay, it works up to that point, and then it doesn't anymore.*'" (**Exp-14**). Similarly, six experts suggested a *comment* function when evaluating test cases, enabling communication between the designer and the executioner. In addition, three experts emphasized the need for *attachments* to display actual results.

Gamification Feedback. The three game-design elements that received the most feedback in the interviews were the *profile*, *leaderboard*, and *point system*. For the profile, experts suggested making the visibility and accessibility optional, "*It's nice to have the option to decide that I don't want people to see that I only have 2 points when all my friends have a hundred."* (**Exp-5**). However, the leaderboard received different feedback from the experts. Two experts proposed expanding the leaderboard by adding more parameters, e.g., renewing at regular intervals. Additionally, four experts suggested adding an unranked status when reaching a certain lower rank. Regarding the game-design element points, several experts recommended widening the allocation of points, e.g., allocating points based on the test case complexity.

Second Iteration: Revising the Conceptual Artifact

The objective of the second design iteration is to refine the concept based on the feedback received from the first evaluation and develop a prototypical implementation. Consequently, after the first design iteration, the core processes for test case design, execution, and evaluation were thoroughly optimized.

The following changes were made based on the interviews:

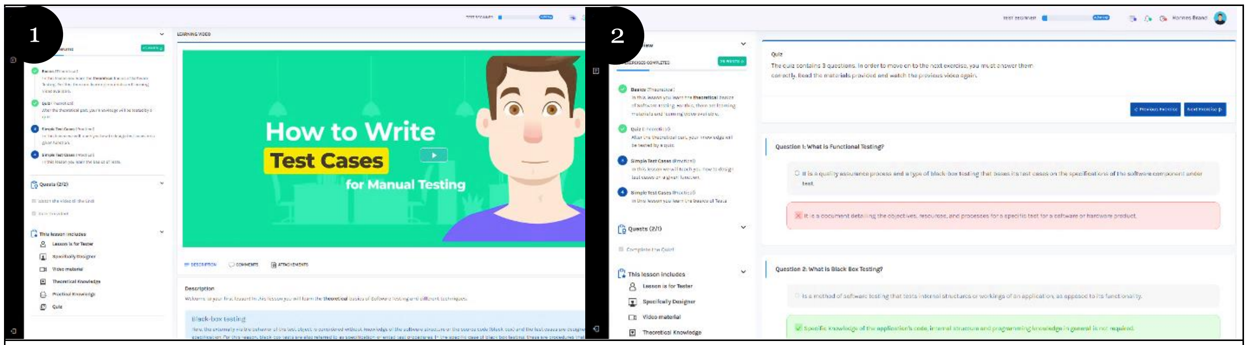
The initial step involves integrating an onboarding system to assist users in navigating the learning platform for the first time. In this onboarding system, we introduce contextual knowledge, and provide step-by-step instructions during the initial interaction, thereby it intakes an informative role and keeps the users engaged with the system. Thus, we add the onboarding *system* ($\rightarrow R_{G9}$) as an motivational requirement.

Regarding the *test case design* ($\rightarrow R_{F3}$), we agreed there is a need to add preconditions to a test case to verify the procedural validity of the test case. Moreover, we also agreed that during the *test case execution* ($\rightarrow R_{F4}$), there is a need for a stepwise comment function, which will improve the traceability of feedback and encourage students to reflect on individual test case steps. However, a significant part of the modifications involves the activities on the *test case evaluation* ($\rightarrow R_{F5}$). As several experts emphasized the significance of a comment and an attachment function, we aim to incorporate them in the next design process. This approach will promote interaction and peer feedback among the students.

As part of the game-design elements, we included an option to configure *profiles* (→R$_{G1}$) as visible or invisible. Using the freedom of choice, we aim to support the needs of the users based on the self-determination theory (Deci & Ryan, 2012). However, the most significant changes were made in how and when the system allocates *points* (→R$_{G5}$). In the conception, points for manually executing test cases were allocated only marginally. We plan to change this by implementing events that reward users for completing certain tasks for the first time, such as creating or executing a test case. This is intended to encourage participation and foster a sense of competence among the students. Additionally, the test case designer will receive points based on the evaluation. Regarding the *leaderboard* (→R$_{G7}$), we agree that users who have been using the learning application for a long time can potentially create more test cases. However, as the application is only used within a course group per semester, the leaderboard remains unchanged. This ensures that each student has an equal chance to improve their rank on the leaderboard.

Second Iteration: Implementing the Black-Box Learning System

Based on the presented results, we implemented the conceptual artifact into a usable web-based learning system based on the overall concept as shown in Figure 3. To facilitate access to the learning application, regardless of time and location, we opted for a client-server architecture, whereby students can access the application via a web browser. We used Debian 11 as an Unix operating system with nginx (version 1.18.0-6) as a web server. The PHP scripting language version 7.4 displays the web content processed by the web server. The web application is built on a *multi-layer Model-View-Controller* (MVC) architectural pattern and employs *Laravel* as a web application framework (Laravel, 2023). To ensure a responsive front end for accessing the application across diverse mobile devices, we utilized the *Bootstrap framework* version 4.6.x (Bootstrap, 2023). Additionally, we used the icon framework *Iconify*, which provides a collection of icons (Iconify, 2023). To guide users through the application during their initial usage, we employed the JavaScript library *ShepherdJS* as an onboarding system (ShepherdJS, 2023).



Quiz

The quiz contains 3 questions. In order to move on to the next exercise, you must answer them correctly. Read the materials provided and watch the previous video again.

Question 1: What is Functional Testing?

It is a quality assurance process and a type of black-box testing that bases its test cases on the specifications of the software component under test.

It is a document detailing the objectives, resources, and processes for a specific test for a software or hardware product.

Question 2: What is Black Box Testing?

Is a method of software testing that tests internal structures or workings of an application, as opposed to its functionality.

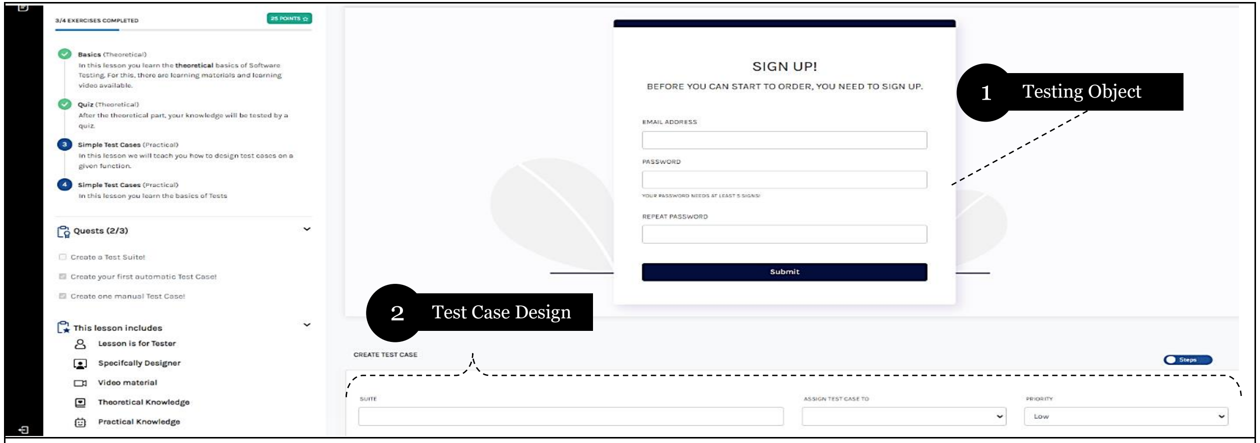
Specific knowledge of the application's code, internal structure and programming knowledge in general is not required.

Figure 5. Screenshot of the Two Exercises

After the students login into the learning platform, they get an overview of available exercises and their progress on a dashboard. During the initial use, the students are guided through the learning platform using an *onboarding system* (→R$_{F1}$) and get stepwise introduced to each exercise. In addition, each exercise is accompanied by *quests* (→R$_{G5}$) providing additional challenges and opportunities for interactions. Moreover, the quests are based on *points* (→R$_{G2}$), serving as orientation for the level of difficulty of a task.

As shown in Figure 5, the first exercise starts with a *video-based learning material* (→R$_{F1}$) based on the Anchored Instructions Design (Cognition and Technology Group, 1993), consisting of a scenario that learners can relate to (Figure 5, mark 1). Thus, the first exercise refers to Bloom's taxonomy *remember* as the users are supposed to recall the facts and concept of functional testing. The second exercise consists of a *quiz* (→R$_{F2}$) to test the theoretical knowledge (Figure 5, mark 2). By implementing a quiz, we aim to foster cognitive understanding, as a quiz requires the student to recall the information presented.

The third exercise addresses the *design of test cases* (→R$_{F3}$) based on testing objects as shown in Figure 6. For this reason, a series of three testing objects were deployed: (1) Registration form, (2) login system, (3) order system to validate the accuracy of orders, which were assigned a different amount of *points* (→R_{G2}), depending on the difficulty. The black-box techniques *boundary value analysis* or *equivalence partitioning testing* are primarily applicable. We added the testing object (mark 1) above a dynamic expanding test case design form (mark 2). The student is supposed to create test cases using the test case design template, thus, the theoretical knowledge is applied to a real testing object, which corresponds to the level *apply* according to Bloom's taxonomy. Moreover, once edge cases were identified, students received specific *badges* (→R_{G6}).



Basics (Theoretical)
In this lesson you learn the **theoretical** basics of Software Testing. For this, there are learning materials and learning video available.

Quiz (Theoretical)
After the theoretical part, your knowledge will be tested by a quiz.

Simple Test Cases (Practical)
In this lesson we will teach you how to design test cases on a given function.

Simple Test Cases (Practical)
In this lesson you learn the basics of Tests

- ☐ Create a Test Suite!
- ☑ Create your first automatic Test Case!
- ☑ Create one manual Test Case!

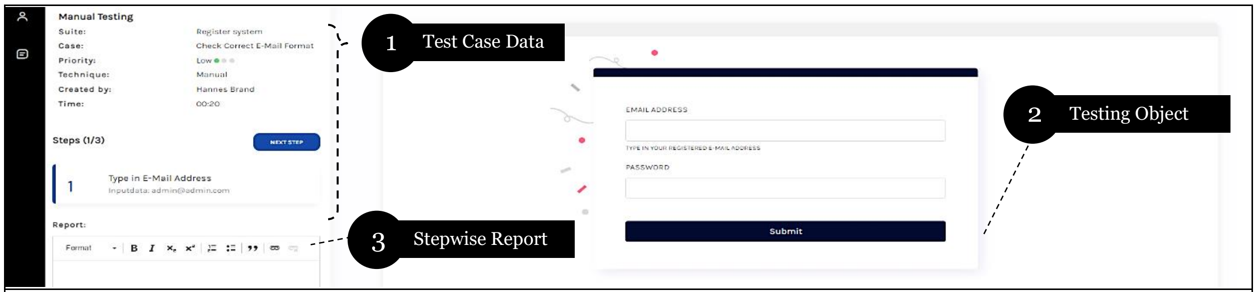
- Lesson is for Tester
- Specifcally Designer
- Video material
- Theoretical Knowledge
- Practical Knowledge

BEFORE YOU CAN START TO ORDER, YOU NEED TO SIGN UP.

YOUR PASSWORD NEEDS AT LEAST 5 SIGNS!

Figure 6. Screenshot of the Test Case Design Process

In the third exercise, students are provided with an overview of all the designed test cases. The overview includes details such as the designer and date of creation for each test case, as well as its *rating* (→R_{G8}) and the possibility to manually execute the test cases. To avoid the misuse of receiving points by designing test cases, students were allocated a score. The better their test cases were rated, the higher the score became. If the score becomes negative, they fall behind on the leaderboard (→R_{G6}) and receive fewer *points* (→R_{G1}).



Manual Testing

Suite:	Register system
Case:	Check Correct E-Mail Format
Priority:	Low ●●●
Technique:	Manual
Created by:	Hannes Brand
Time:	00:20

Steps (1/3)

1 Type in E-Mail Address
Inputdata: admin@admin.com

Report:

TYPE IN YOUR REGISTERED E-MAIL ADDRESS

Figure 7. Screenshot of the Test Case Execution Process

Figure 7 displays the *execution of a test case* (→R_{F4}). The student can complete each step (mark 1) and apply the input instruction directly to the testing object (mark 2). The test case execution aims to analyze the test case design and check the correctness of the test case based on the testing objects. If there is an incorrect step, a report can be submitted (mark 3). Moreover, by analyzing the results, we aim to enhance the cognitive level of *analyze* according to Bloom's Taxonomy. After executing a test case, it is possible to provide a *rating* (→R_{G8}) using a five-star rating system, along with adding attachments and feedback. *Evaluating the test case* (→R_{F5}) and appropriate action-taking, such as categorizing a test case as failed, corresponds to the cognitive level of *evaluation* according to Bloom's Taxonomy.

In addition, the application provides a profile. The *profile* (→R_{G1}) of each student contains key data, contact details, and an individual user description. However, the profile also serves as an overview of the student's performance. Accordingly, achieved *points* (→R_{G2}), the *level* (→R_{G3}), *badges* (→R_{G6}), and a *leaderboard* (→R_{G8}) are displayed. However, it is possible to configure the profile, e.g., to deactivate participation on the leaderboard, anonymize involvement in the learning system, or deactivate the profile for public access.

Second Iteration: Evaluation of the Implementation

To evaluate students' perceptions regarding the concept of the teaching of black-box testing competencies and game-design elements, we conducted a quantitative analysis using a questionnaire. The questionnaire consisted of an introduction to black-box testing technique and three scenarios, which the students had to perform before answering the questionnaire: (1) designing a test case, (2) executing a test case manually, and (3) checking on reports and ratings. The respondents also could provide feedback by open text fields.

In summary, **31 information systems students** completed the questionnaire, consisting of three blocks:

1. The first block addressed students' *testing experience* and their *affinity for IT systems* based on a 6-point Likert scale (strongly disagree [1] - strongly agree [6]) (ATI Scale, 2022).
2. In the second block, we assessed students' perceived *usefulness of features* and *game-design elements* as well as their *intention of use* and *perceived learning success* based on a 6-point Likert scale (strongly disagree [1] - strongly agree [6]), which provides a granular range of responses.
3. Lastly, the fourth block consisted of the *user experience questionnaire* (UEQ+ Online, 2022) based on a 7-point semantic differential scale (from [-3] to [+3]).

To gain insight into the dataset, we applied *descriptive statistics* and a 2-tailed *Spearman's correlation analysis*. In the first block, the results showed that the respondents had an average *affinity for technology* (M = 4.06). Moreover, we were able to confirm that students predominantly have a low level of experience in testing, as the result shows that the students indicated low experience with *testing techniques* and *testing tools* (M = 3.1). Similarly, students had little experience with *automated testing* (M = 3.0) or *manual testing* (M = 3.0). In contrast, the students are aware that testing is *crucial in software development* (M = 5.5).

Furthermore, Figure 8 summarizes the results of the second block regarding the usefulness of the individual functionalities and game-design elements and lastly, the perceived usability of individual functionalities.

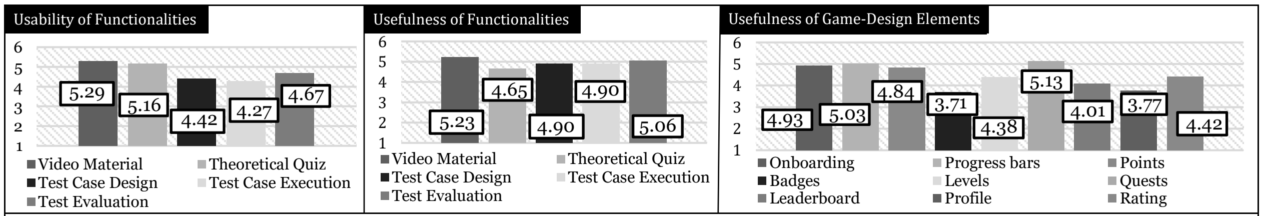


Figure 8. Overview of the Evaluation Results 1

In general, each functionality was perceived positively in terms of its usefulness and usability. The highest usefulness perceived was the *video-based learning material* (M = 5.23), grounding the basis for gaining theoretical knowledge in a realistic scenario. While the three processes of *design* (M = 4.90), *execution* (M = 4.90), and *evaluation* (M = 5.06) were considered useful, they indicate lower perceived usability. In comparison, the perceived *usability of the test case design* (M = 4.42), *execution* (M = 4.27), and *evaluation* (M = 4.67) were lower than the *video-based learning material* (M = 5.29). This can be due to students' inexperience in terms of testing procedures and techniques. Moreover, one student commented that the quiz was designed to be too simple, resulting in being underchallenging. The usefulness of the game-design elements, e.g., the *onboarding system* (M = 4.93), *point system* (M = 4.84), *level system* (M = 4.38), and *quests* (M = 5.13) have been perceived as useful. Whereas *badges* (M = 3.71), and the *profile* (M = 3.77) were perceived as less useful.

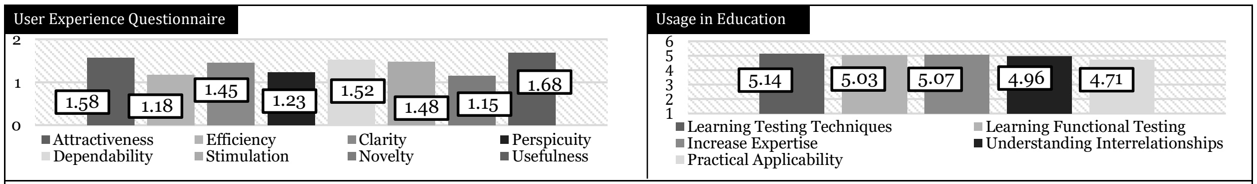


Figure 9. Overview of the Evaluation Results 2

The results of the UEQ+ are calculated based on the average scale (Figure 9). In general, the achieved score of the UEQ+ aspects are positive but differ noticeably. Among the eight aspects, *usefulness* (M = 1.68), *attractiveness* (M = 1.58), *dependability* (M = 1.52), *clarity* (M = 1.45), and *stimulation* (M = 1.48) received the highest score. However, the aspects *perspicuity* (M = 1.23), *efficiency* (M = 1.18), and *novelty* (M = 1.15) achieved the lowest score. According to this, the students perceived the learning system as less transparent, efficient, and innovative but attractive and interesting way of teaching black-box testing. Thereby, it should be noted that a large proportion of the students perceived the application as an effective approach for learning testing techniques, e.g., boundary-value analysis (M = 5.14). The students also see the potential for improving the understanding of *interrelationships between test processes* (M = 4.71). In addition, respondents stated, they strongly envisioned using this application in a course or lecture to *increase the expertise* (M = 5.14) in black-box testing, indicating the utility of the design implementation.

Additionally, the correlation analysis shows that there is a correlation between *technical affinity* and perceived usefulness of the *video material* ($\rho = .384$, $p < 0.05$) and *test execution* ($\rho = .487$, $p < 0.01$). Therefore, we assume that technical understanding is required to comprehend the realistic scenario in videos. In this context, it is also evident that the perceived usefulness of the video correlates with *test design* ($\rho = .499$, $p < 0.05$), *execution* ($\rho = .523$, $p < 0.05$), and *evaluation* ($\rho = .523$, $p < 0.05$). This could mean that the video is an essential component for conveying fundamental knowledge. Regarding game-design elements, the *onboarding* seems significant for the *test case design* ($\rho = .499$, $p < 0.05$) and *test case execution* processes ($\rho = .583$, $p < 0.05$). It is also interesting to note that although the respondents found badges less useful, there was a positive correlation between *badges* and *test case design* ($\rho = .401$, $p < 0.05$) and *execution* ($\rho = .412$, $p < 0.05$), indicating that badges for discovering edge-cases or errors in test cases through the execution process can have a positive impact. Moreover, between the *rating system* and the *execution* process ($\rho = .449$, $p < 0.05$) exists a positive correlation. The correlation analysis also indicates a correlation between the use of the application in educational settings for imparting *practical knowledge* and the exercises *test case design* ($\rho = .484$, $p < 0.01$) and *test case execution* ($\rho = .442$, $p < 0.01$) processes.

Overall, the evaluation results indicate that the learning system has the potential to support students in acquiring knowledge about functional testing, despite students' limited prior experience in testing.

Field Test – Data Analysis and Results

Since we could not identify any significant changes to the software testing process, the design process was completed. Therefore, we decided to conduct a field test at a German university. The field test is aimed at utilizing the system in a real teaching environment to measure its practical usefulness as well as effectiveness.

For this, the field test consisted of three parts:

1. We gathered information about students' level of knowledge in software testing, technical affinity, and perceived competence in software testing.
2. In addition, the students could participate in a pre-test optionally.
3. Students could interact freely with the application for at least 30 minutes (max. 60 minutes).
4. Students had to participate in a post-test to measure learning outcomes.

To measure the overall experience of the application, we included the Cognitive Load Theory (**CLT**) by Klepsch (2020), consisting of 8 items based on a 7-point-Likert scale (absolutely wrong [1] – absolutely right [7]). The CLT is an educational framework that describes the amount of mental effort required to understand and perform a task. Sweller's (2010) CLT postulates that the human working memory has a limited capacity, and when this capacity is exceeded, learning becomes less efficient and less effective. For this, the CLT consists of three types of loads: Intrinsic, extraneous, and germane. First, the component *intrinsic cognitive load* (**ICL**) describes the complexity of the learning task. Second, the *extraneous cognitive load* (**ECL**) refers to any additional cognitive demands caused by suboptimal instructional design. Third, the *germane cognitive load* (**GCL**) refers to the mental effort required to actively engage with the task and arises from the learner's understanding of the learning material (Klepsch, 2020). Last, to determine the interrelationship in fostering a learning process, we used the *Intrinsic Motivation Inventory* based on Deci and Ryan (2011) with a 5-point-Likert scale (strongly disagree [1] – strongly agree [5]).

In summary, **59 undergraduate students** with IS background participated in the field test. To analyze the results of the field test, we first conducted a *t-test* to compare the results of the pre- and post-tests. Moreover, we conducted a *Spearman's correlation analysis* to uncover interrelationships.

Results in Pre- and Post-tests

Before we started the pre-test, we asked the students to self-esteem their developing and testing skills. The results indicated that they had a modest level of understanding regarding the fundamentals of **software development** (M=2.80) and **software testing** (M = 2.52). Furthermore, their self-reported level of expertise in more specialized domains such as **test case design** (M = 1.85), as well as **manual** (M = 2.0), and **automatic** (M = 1.76) testing, was limited, which was in line with the expected limited experience.

The pre-test and post-test consisted of eight multiple-choice questions and two practical tasks, with a maximum score of 14 points. In addition, 16 students participated in the optional pre-test. The results of the pre-test and post-test show that the participating students scored lower on the **pre-test** (M = 7.44, S = 2.73) than on the **post-test** (M = 10.50, S = 2.36). Comparing both results, the t-test yielded a t-value of t(15) = -2.75 and a p-value of .009. At a significance level of 0.05, we can conclude that there is a significant difference between the mean scores of the pre-test and the post-test. The effect size, as indicated by the correlation value (|r| = .704) is considered as *medium* according to Cohen (2013), indicating that the use of the learning application led to a significant increase in scores compared to before using the learning platform iTest.

Results of Cognitive Load Theory

The study found that the learning application was easy to use and understand for students, reducing the **ICL** with a mean value of 3.48. This suggests that the application was designed in a way that minimized the inherent complexity of the task, making it more accessible for students. However, the **GCL** (M = 4.88) indicates that students required more time and effort to understand important aspects of the task. This suggests that the instructional design may not be optimized, as students experienced a higher cognitive load related to constructing knowledge and problem-solving. On the other hand, the **ECL** (M = 3.31) is indicating that distractions were minimal. This suggests that the instructional design was effective in minimizing irrelevant information or design elements that could have contributed to a higher cognitive load. The focus remained on the relevant aspects of black-box testing. Overall, the cognitive load experienced by students was moderate but could be improved by maximizing GCL and providing necessary information while minimizing *ECL* to avoid distractions.

Results of Intrinsic Motivation Inventory

Based on the results, students showed positive **enjoyment/interest** (M = 4.02) in performing the test activities. Moreover, regarding the **perceived competence** (M = 3.43) and **perceived choice** (M = 3.55), students had a moderate perception of their abilities during the testing activities and the freedom to make their own decisions. The perceived **effort/importance** (M = 3.41) indicates that the testing activities were not particularly balanced, while the **pressure/tension** (M = 2.88) was rather low during the testing activities. Lastly, the results indicate a moderate perception of **relatedness** (M = 3.20).

Moreover, the correlation analysis revealed possible factors that could have an impact on the motivation dimensions. First, the results indicate a positive correlation between the *perceived knowledge about software testing* and the **perceived interest/enjoyment** (ρ = .299, p < 0.05). Second, the results also indicate a positive correlation between *interest/enjoyment* and *knowledge in practice* (ρ =.505, p < 0.01). This implies that as students' knowledge about software testing increases, their level of interest and enjoyment in the subject also tends to increase. Overall, this result is also supported as there is a positive correlation between the GCL and students' perceived *interest/enjoyment* of the learning experience (ρ = .378, p < 0.01). Accordingly, the results may indicate that a higher GCL, i.e., mental effort, leads to greater students' interest, which can have a positive impact on motivation. Additionally, we also found a negative correlation between *ECL* and the *perceived interest* for the application (ρ = -.370, p < 0.05), indicating that the learning platform is not optimized in minimizing distracting elements.

Regarding the **perceived competence**, correlation analysis reveals that there is a correlation between the perceived competence and general *knowledge about software testing* (ρ = .627, p < 0.05). Accordingly, an increase in knowledge through the learning platform can lead to a higher experience of competence.

In summary, the learning platform contributes to the knowledge of black-box testing. Based on the cognitive load theory it was found that the application was easy to understand but contained minimal distractions.

Discussion and Conclusion

The study aimed to design a learning platform for teaching black-box testing by using practical testing objects. Thus, we provide an approach to shift from a theory-based approach to a practice-based approach. Especially, by using game-design elements we aimed to improve the quality of designed test cases and encourage students to execute the test cases manually. Thus, by presenting a situated artifact in a learning context that contributes to the knowledge base, our design provides a level 1 DSR contribution. Based on the evaluation results regarding the core functional requirements and game-design elements, we derived six design principles according to Gregor et al. (2020) as shown in Table 3.

#	Description	RQ
DP1	For developers designing a gamified learning platform to foster students' black-box competencies, provide a function for lecturing fundamentals of black-box testing theoretically, so students can acquire knowledge about black-box testing by first learning and memorizing the basics, as per Bloom's taxonomy.	R_{F1} R_{F2}
DP2	For developers designing a learning platform to foster students' black-box competencies, provide a function for designing a test case based on a given object, thereby applying theoretical knowledge.	R_{F3}
DP3	For developers designing a learning platform to foster students' black-box competencies, implement a function to execute and evaluate test cases and thus, improve students' black-box competencies by reflecting and judging.	R_{F4} R_{F5}
DP4	For developers designing a learning platform to foster students' black-box competencies, implement a function for providing feedback to report errors, ambiguities, or bugs.	R_{F5}
DP5	For developers designing a learning platform to foster students' black-box competencies, implement a function to visualize the current progress, promote activity, and foster self-control and self-monitoring in the system, thus providing immediate feedback.	R_{G2} R_{G3} R_{G4}
DP6	For developers designing a learning platform to foster students' black-box competencies, implement a function to get introduced to the core functionalities and possibilities in the system to assist with first-time usage thus, ensuring a structured introduction.	R_{G9}

Table 3. Derived Design Principles

Our results indicate that the application creates a foundation for increasing black-box testing skills through exercises. Realistic instructional videos should be used as a foundation to establish practical relevance (**DP1**). Moreover, based on practical knowledge, there should be an opportunity to practically design (**DP2**) and execute tests (**DP3**) using testing objects, thereby utilizing the gained knowledge. In addition, the reflection competencies are to be promoted by a reporting system (**DP4**). Moreover, improving the student's sense of competence, motivational elements (**DP5**), e.g., progress bars, points, and performance graphs indicate students' progress throughout the course. However, to avoid difficulties during the first use and information overload, the process should be supported by an onboarding (**DP6**).

From theoretical to practical testing objects. The present approach intends to reduce the deficiency in the education of software testing knowledge as testing processes are rarely taught in IS courses. However, teaching test competencies is essential in software development. This learning application offers an approach that combines theory-based teaching with practice-oriented testing objects. Starting with the video-based learning material, which was particularly positively evaluated, up to the use of theoretical knowledge on practical objects. In addition, with the transition from theory to practice, the learning platform can increase students' awareness regarding testing when test techniques are used on real objects.

Coverage of an entire testing process. The results show that especially the functionalities for test design, execution, and evaluation were considered useful. Interrelationships can be better understood by working through the processes one after another. As a result, students should not just learn one process unilaterally, but also learn how the processes work together.

Software testing training. Despite the need to foster software testing education in an instructional context, there is a high demand for software testing training in the industry (Aniche et al., 2019). The provided solution and design principles are applicable to the educational as well as the industrial context (Garousi et al., 2020). Thus, the present research also provides an approach for industrial use.

Nevertheless, we acknowledge that making software testing courses practical and industry-aligned is important but challenging (Aniche et al. 2019). The presented application covers basic functionalities and may not sufficiently demonstrate significant industry benefits. Possibly, students were overchallenged by the test case design, execution, and evaluation or the novelty phenomenon of gamification. Dividing students into multiple groups may also provide additional insights into the efficiency of gamification.

References

Alhroob, A., Dahal, K., & Hossain, M. (2012). *Software test case generation from system models and specification. Use of the UML diagrams and High Level Petri Nets models for developing software test cases* [Dissertation, University of Bradford]. RIS. https://hdl.handle.net/10454/5453

Anderson, L. W., & Krathwohl, D. R. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A revision of Bloom's taxonomy of educational objectives*. Longman.

Aniche, M., Hermans, F., & van Deursen, A. (2019). Pragmatic software testing education. In *SIGCSE '19, Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery. https://doi.org/10.1145/3287324.3287461

ATI Scale. (2022). *Affinity for Technology Interaction Scale*. https://ati-scale.org/

Blanco, R., Trinidad, M., Suárez-Cabal, M. J., Calderón, A., Ruiz, M., & Tuya, J. (2023). Can gamification help in software testing education? Findings from an empirical study. *Journal of Systems and Software, 200*, 111647. https://doi.org/10.1016/j.jss.2023.111647

Bloom, B. S. (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals. Taxonomy of Educational Objectives: The Classification of Educational Goals*. Longman.

Bootstrap. (2023). *Build fast, responsive sites with Bootstrap*. https://getbootstrap.com/

Clarke, P. J., Davis, D., King, T. M., Pava, J., & Jones, E. L. (2014). Integrating testing into software engineering courses supported by a collaborative learning environment. *ACM Transactions on Computing Education (TOCE), 14*(3), 1–33. https://doi.org/10.1145/2648787

Cognition and Technology Group (1993). Anchored instruction and situated cognition revisited. *Educational Technology*(33), Article 3, 52–70. https://www.jstor.org/stable/44427992

Cohen, J. (2013). *Statistical Power Analysis for the Behavioral Sciences*. Elsevier Science. https://doi.org/10.4324/9780203771587

Costa, I., & Oliveira, S. (2019). A Systematic Strategy to Teaching of Exploratory Testing using Gamification. In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering* (pp. 307–314). Science and Technology Publications.

Deci, E. L., & Ryan, R. M. (2011). Levels of analysis, regnant causes of behavior and well-being: The role of psychological needs. *Psychological Inquiry*(1), 17–22. https://doi.org/10.1080/1047840X.2011.545978

Deci, E. L., & Ryan, R. M. (2012). Self-determination theory. In *Handbook of theories of social psychology, Vol. 1* (pp. 416–436). sage publications ltd. https://doi.org/10.4135/9781446249215.n21

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining" gamification". In *MindTrek '11, Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. Association for Computing Machinery.

Elbaum, S., Person, S., Dokulil, J., & Jorde, M. (2007). Bug hunt: Making early software testing lessons engaging and affordable. In *29th International Conference on Software Engineering (ICSE'07)*.

Garcia, F., Pedreira, O., Piattini, M., Cerdeira-Pena, A., & Penabad, M. (2017). A framework for gamification in software engineering. *Journal of Systems and Software*, *132*, 21–40.

Garousi, V., Rainer, A., Lauvås Jr, P., & Arcuri, A. (2020). Software-testing education: A systematic literature mapping. *Journal of Systems and Software, 165*. https://doi.org/10.1016/j.jss.2020.110570

Gregor, S., Chandra Kruse, L., & Seidel, S. (2020). Research perspectives: the anatomy of a design principle. *Journal of the Association for Information Systems*, *21*(6), 2. https://doi.org/10.17705/1jais.00649

Haas, R., Elsner, D., Juergens, E., Pretschner, A., & Apel, S. (2021). How can manual testing processes be optimized? developer survey, optimization guidelines, and case studies. In D. Spinellis, G. Gousios, M. Chechik, & M. Di Penta (Eds.), *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1281–1291). ACM. https://doi.org/10.1145/3468264.3473922

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems, 19*(2), 4. https://aisel.aisnet.org/sjis/vol19/iss2/4

Hooda, I., & Chhillar, R. S. (2015). Software test process, testing types and techniques. *International Journal of Computer Applications*(111), Article 13. https://doi.org/10.5120/19597-1433

Hunicke, R., Leblanc, M., & Zubek, R. (Eds.) (2004). *MDA: A formal approach to game design and game research.* : Vol. *1*. San Jose, CA.

Hynninen, T., Knutas, A., & Kasurinen, J. (Eds.) (2019). *Designing Early Testing Course Curricula with Activities Matching the V-Model Phases*. IEEE. https://doi.org/10.23919/MIPRO.2019.8757033

Iconify. (2023). *Freedom to choose icons*. https://iconify.design/

Jain, C. R., & Kaluri, R. (2015). Design of automation scripts execution application for selenium webdriver and test NG framework. *ARPN J Eng Appl Sci*, *10*(6), 2440–2445.

Jesus, G. M. de, Ferrari, F. C., & Porto, D. d. P. (2018). Gamification in software testing: A characterization study. In *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing*. https://doi.org/10.1145/3266003.3266007

Jesus, G. M. de, Paschoal, L. N., Ferrari, F. C., & Souza R. S. Simone (Eds.) (2019). *Is it worth using gamification on software testing education? an experience report*.

Kapil Singi (Ed.) (2020). *Are software engineers incentivized enough? An outcome based incentive framework using tokens*. IEEE. https://doi.org/10.1109/IWBOSE50093.2020.9050262

Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons.

Kaprocki, Z., Pekovic, V., & Velikic, G. (Eds.) (2015). *Combined testing approach: Increased efficiency of black box testing*. IEEE. https://doi.org/10.1109/CEWS.2015.7867160

Klepsch, M. (2020). *Differenzierte Messung kognitiver Belastung beim Lernen im Rahmen von Instruktionsdesignfragestellungen* [Doctoral Dissertation]. Universität Ulm.

Kris, H., & Heider, J. (Eds.) (2020). *Raising Security Awareness on Mobile Systems through Gamification*. https://doi.org/10.1145/3424954.3424958

Krutz, D. E., Samuel A. Malachowsky, & Thomas Reichlmayr. (2014). *Using a Real World Project in a Software Testing Cours: Proceedings of the 45th ACM Technical Symposium on Computer Science Education ; March 5 - 8, 2014, Atlanta, Georgia, USA*. ACM.

Laravel. (2023). *The PHP Framework for Web Artisans*. https://laravel.com/

Ottfutt, J., Li, Nan, Ammann, Paul, & Xum Wuzhi (Eds.) (2011). *Using abstraction and Web applications to teach criteria-based test design*. IEEE. https://doi.org/10.1145/2538862.2538955

Parizi, R. M. (Ed.) (2016). *On the gamification of human-centric traceability tasks in software testing and coding*. IEEE. https://doi.org/10.1109/SERA.2016.7516146

Paschoal, L. N., Oliveira, B. R. N., Nakagawa, E. Y., & Souza, S. R. S. (2019). Can we use the Flipped Classroom Model to teach Black-box Testing to Computer Students? In A. B. Albuquerque & A. L. B. de Paula Barros (Eds.), *Proceedings of the XVIII Brazilian Symposium on Software Quality* (pp. 158–167). ACM. https://doi.org/10.1145/3364641.3364659

Porto, D. d. P., Jesus, G. M. de, Ferrari Fabiano, C., Fabbri, Camargo, Sandra, & Ferraz, P. (2021). Initiatives and challenges of using gamification in software engineering: A Systematic Mapping. *Journal of Systems and Software*, *173*. https://doi.org/10.1016/j.jss.2020.110870

Sailer, M. (2016). Wirkung von Gamification auf Motivation. In *Die Wirkung von Gamification auf Motivation und Leistung* (pp. 97–126). Springer. https://doi.org/10.1007/978-3-658-14309-1

Sánchez-Gordón, M.-L., & Moreno, L. (2014). Toward an Integration of Web Accessibility into Testing Processes. *Procedia Computer Science*, *27*, 281–291. https://doi.org/10.1016/j.procs.2014.02.031

Santos, I., Mori, A., & Souza R. S. Simone (Eds.) (2021). *Using an Incremental Testing Strategy to Improve Students' Perception of Software Quality*. SBC. https://doi.org/10.5753/wei.2021.15909

Scatalon, L. P., Prates, J. M., Souza, D. M. de, Barbosa, E. F., & Garcia, R. E. (2017). Towards the Role of Test Design in Programming Assignments. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T)*. https://doi.org/10.1109/CSEET.2017.34

Sharif, F., & Hemmati, H. (Eds.) (2018). *Investigating nlp-based approaches for predicting manual test case failure*. IEEE. https://doi.org/10.1109/ICST.2018.00038

ShepherdJS. (2023). *Guide your users through a tour of your app*. https://shepherdjs.dev/

Sweller, J. (2010). Cognitive load theory: Recent theoretical advances. *Cognitive Load Theory*, 9–47. https://doi.org/10.1017/CBO9780511844744.004

Thakur, A., & Sharma, G. (Eds.) (2018). *Neural Network Based Test Case Prioritization in Software Engineering*. Springer. https://doi.org/10.1007/978-981-13-3143-5_28

UEQ+ Online. (2022). https://www.ueq-online.org/

Valle, P. H. D., Ricardo Ferreira, V., & Hernandes, E. C. M. (Eds.) (2020). *Does Gamification Improve the Training of Software Testers? A Preliminary Study from the Industry Perspective ∗*.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, xiii–xxiii. https://www.jstor.org/stable/4132319

Yujian, F., & Clarke, P. J. (Eds.) (2016). *Gamification-based cyber-enabled learning environment of software testing*. https://doi.org/10.18260/p.27000